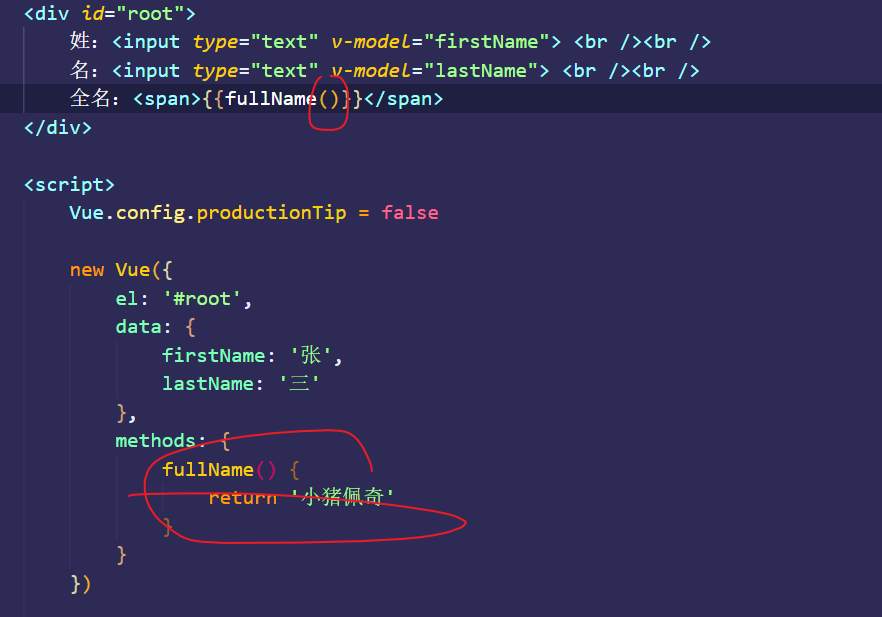
读取一个对象里不存在的属性值是不会报错的，因为是undefin

事件小括号加也行不加也行，但是插值语法里要加



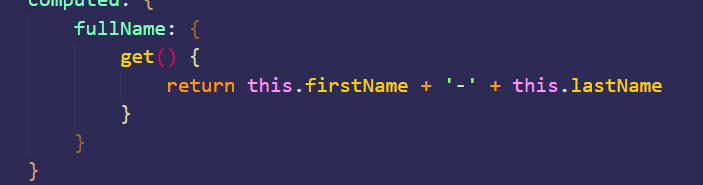
小括号是必须加的，因为不是绑定事件

fullName函数里的this不是window，而是vue实例对象，因为不是箭头函数

改一次fullName就会调用，vue会重新解析模板 插值语法里写方法，方法一定会重新调一遍

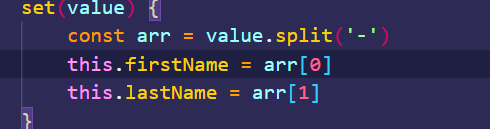
计算属性配置项 computed

计算属性有缓存，get只会调一次



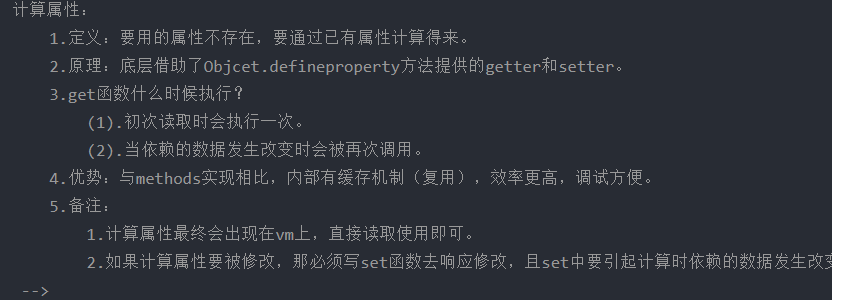
fullName是对象

set不是必须写的 如果说fullName会被人改，就得写set

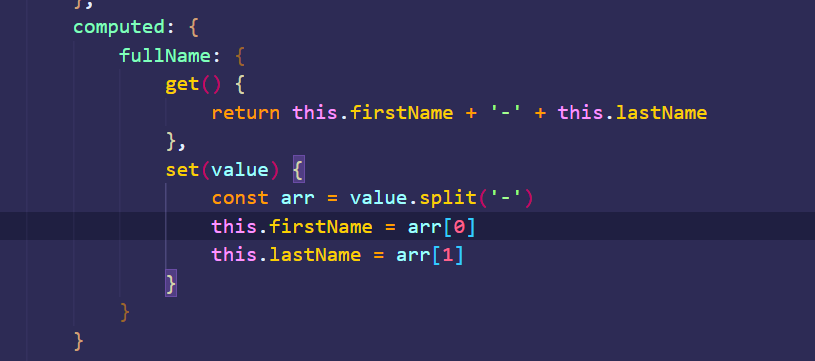


按照-分成两部分放到arr数组里

注意不要写成箭头函数

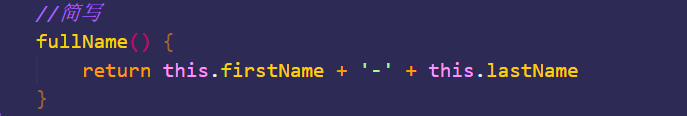


计算属性完整写法



简写 只读不改就能简写也就是说没有set

function当get用



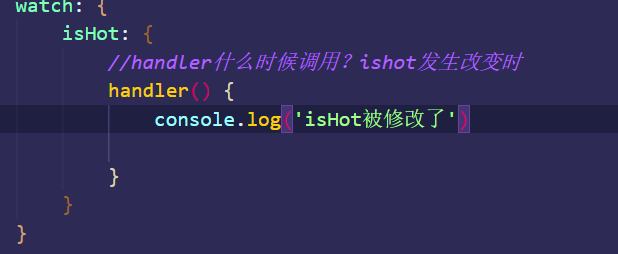
alert是window的



监视配置 watch

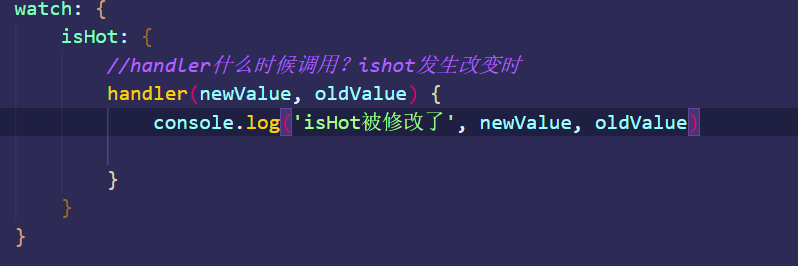
写法跟计算属性一样，也是个对象，比如要监视ishot 就写成 ishot:{}

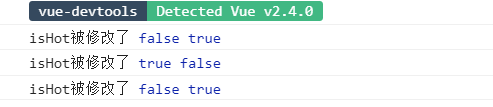
但是{}写的不是get

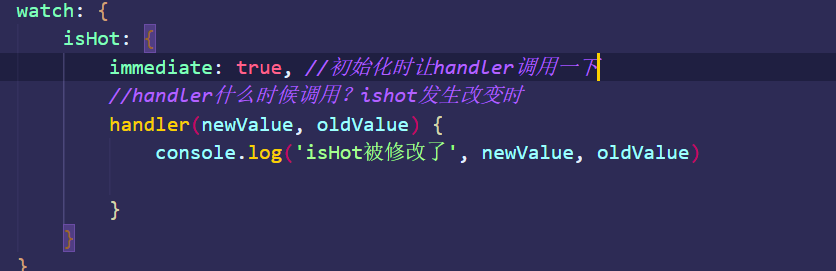


handler不仅能够发现被修改了，还能把修改前的值给你 也会把修改后的值给你

靠参数 ，一个叫newValue,另外一个叫oldVaule







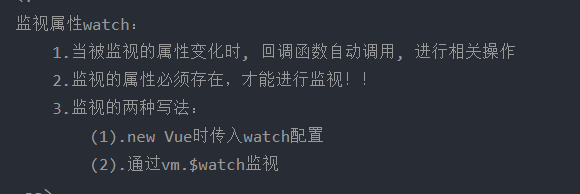
ishot是被监视对象

还有一种监视方法就是不写watch配置

通过vm也能实现一个监视 前提是实例已经创建完了



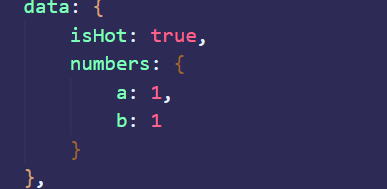
类似于data的第二种写法



回调函数指的是handler

关于第二点，如果没有的，但是不会报错

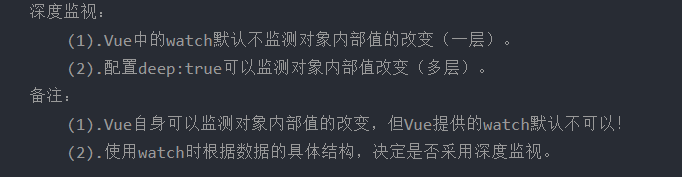
深度监视 监视多级结构中某个属性的变化



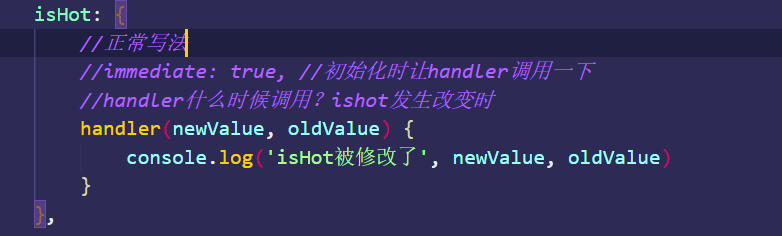


加一个deep就能监视number里面的了

也就是深度监视开启了

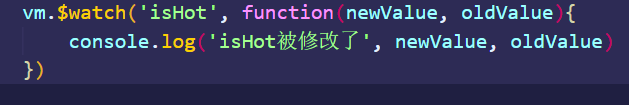


监视的简写：不需要deep和immediate



简写法



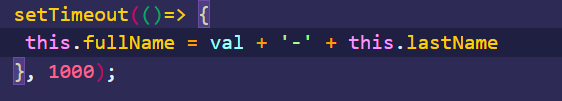


反正vue里不要写箭头函数

所有vue管理的函数都写成普通函数

watch不仅要监视姓，还要监视名

需求：改完姓之后名字一秒后才变，这个时候只能用watch实现

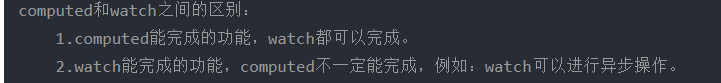


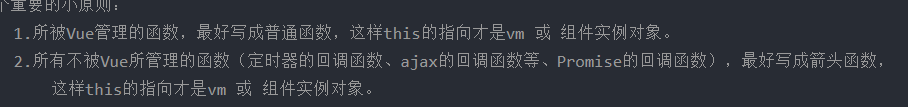
注意，这里的定时器是箭头函数了

箭头函数定义在watch对象内，this指向watch 也就是说，用了箭头函数，this会往外指

而外层函数是watch是普通函数，this所指的就是vm实例

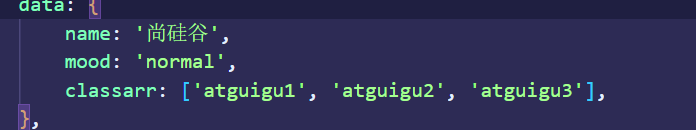
计算属性里不能开启异步任务，而watch可以 异步计算的时候使用watch

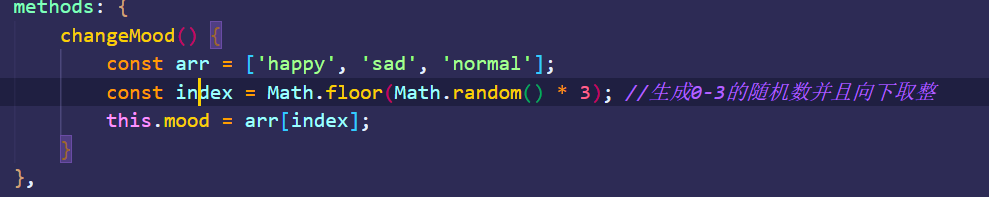




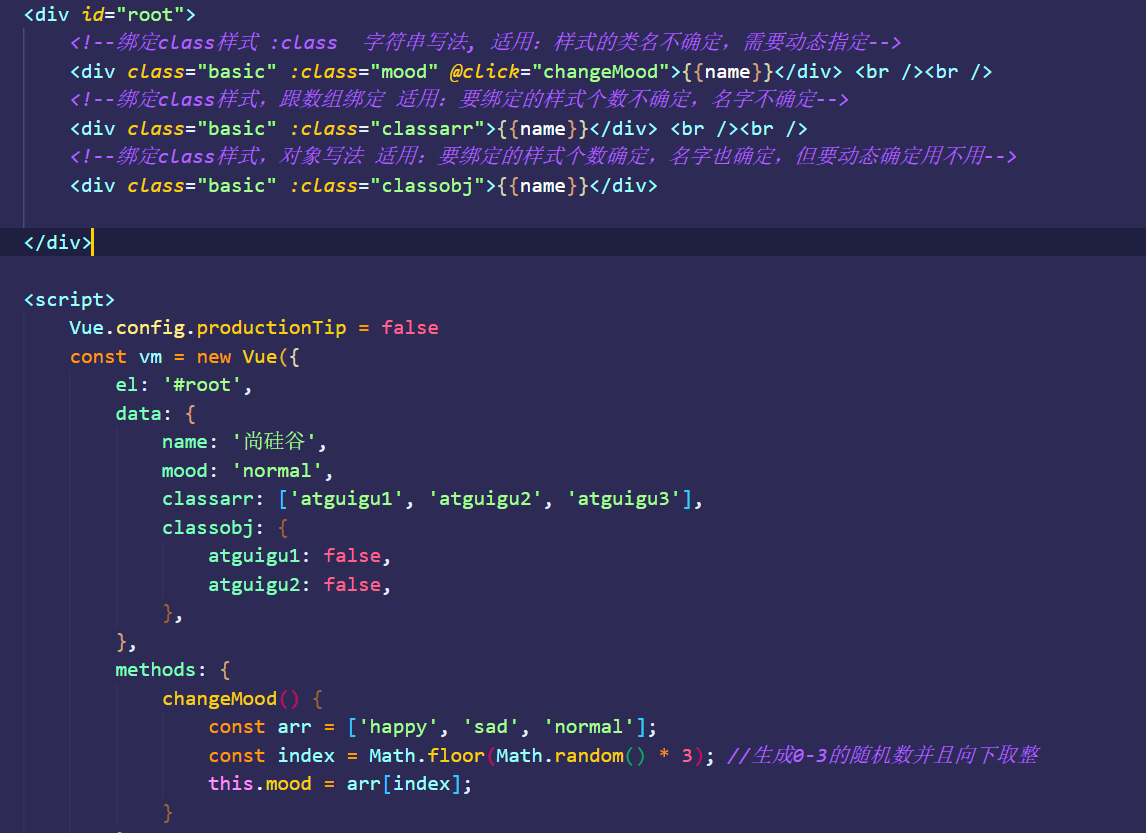


数组



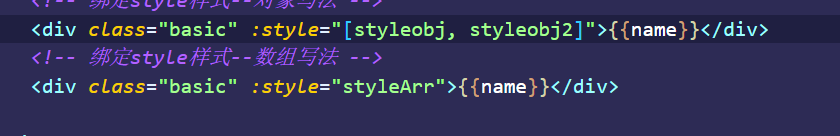


三种颜色随机切换，其实这个随机函数可以再改一改



能这么写





这样写就隐藏了这句

false也可以写成任何表达式

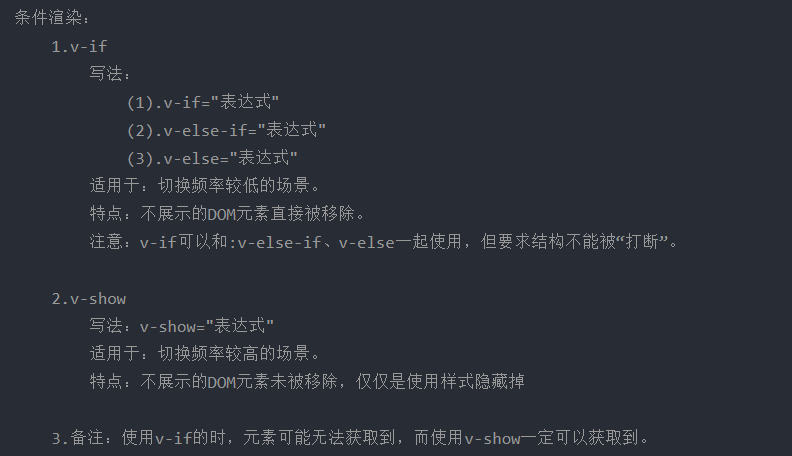
v-if结构都干掉了，但是v-show是用displa-none实现的

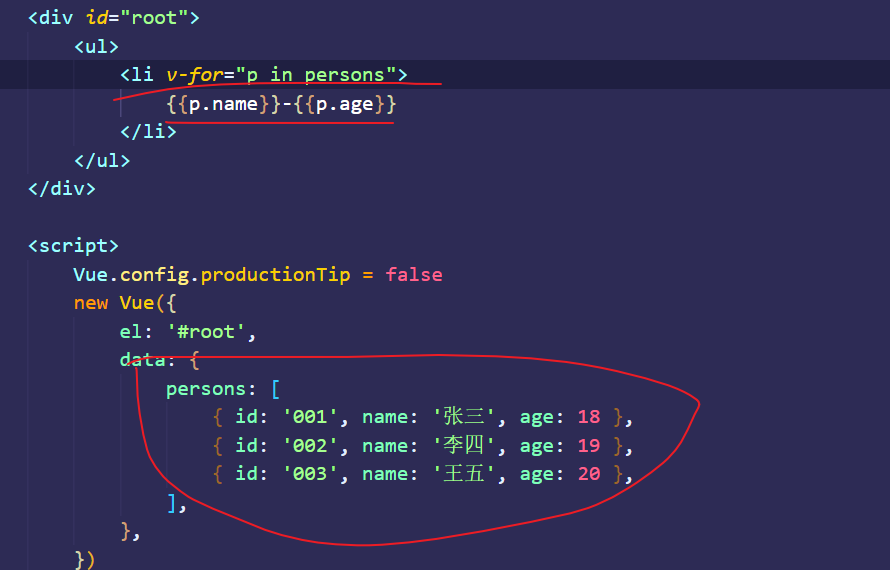
v-if比较损耗性能，频率高的用v-show，低的用v-if

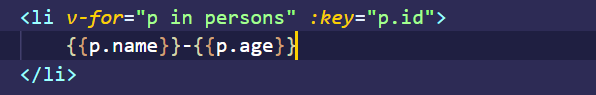
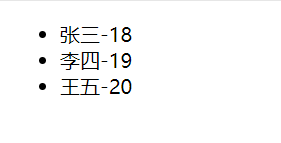
v-if v-else-if v-else与if else一样

v-else后面可以不用写条件，因为前面的都不满足，所以v-else后面写条件没有意义

template最大的特点就是不影响结构

+

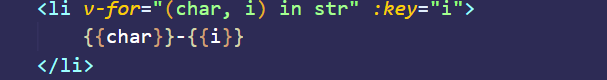


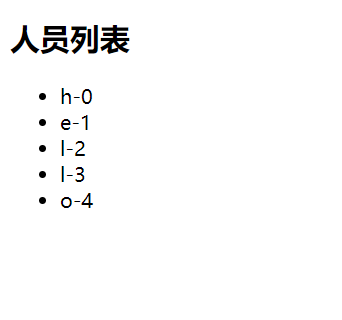


让每个li有个唯一标识 要绑定key哦

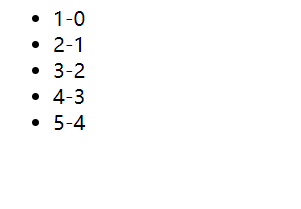
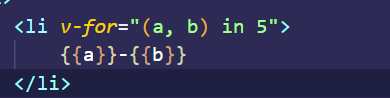
类似于foreach循环

遍历字符串，i相当于下标

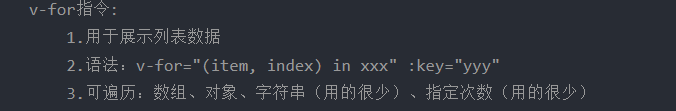




遍历指定次数



a从1开始计数，b是下标



面试题：react、vue中的key有什么作用？（key的内部原理）

1. 虚拟DOM中key的作用：

key是虚拟DOM对象的标识，当数据发生变化时，Vue会根据【新数据】生成【新的虚拟DOM】,

随后Vue进行【新虚拟DOM】与【旧虚拟DOM】的差异比较，比较规则如下：

2.对比规则：

(1).旧虚拟DOM中找到了与新虚拟DOM相同的key:

①.若虚拟DOM中内容没变, 直接使用之前的真实DOM！

②.若虚拟DOM中内容变了, 则生成新的真实DOM，随后替换掉页面中之前的真实DOM。

(2).旧虚拟DOM中未找到与新虚拟DOM相同的key:

创建新的真实DOM，随后渲染到到页面。

3. 用index作为key可能会引发的问题：

1. 若对数据进行：逆序添加、逆序删除等破坏顺序操作:

会产生没有必要的真实DOM更新 ==> 界面效果没问题, 但效率低。

2. 如果结构中还包含输入类的DOM：

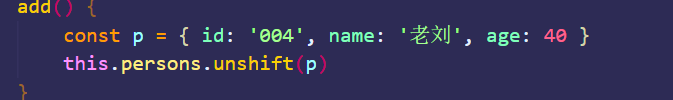
会产生错误DOM更新 ==> 界面有问题。

4. 开发中如何选择key?:

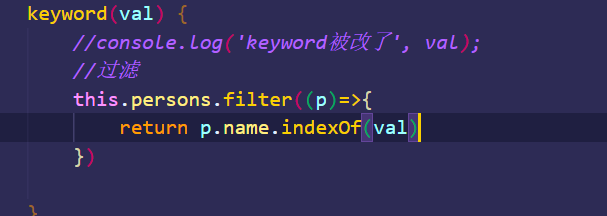
1.最好使用每条数据的唯一标识作为key, 比如id、手机号、身份证号、学号等唯一值。

2.如果不存在对数据的逆序添加、逆序删除等破坏顺序操作，仅用于渲染列表用于展示，使用index作为key是没有问题的。

-->



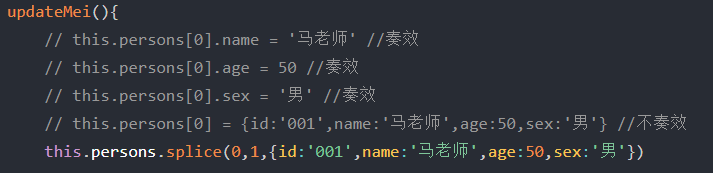
把p放在persons数组的前面



过滤器，其中indexOf(VAL)是判断字符串里是否包含val

不包含返回-1，包含返回下标

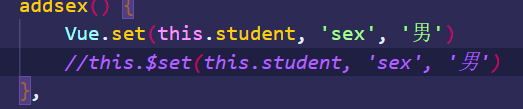
filter过滤完后不改变原数组，filter返回新数组



第四行那么写开发者工具监测不到 但实际上数据已经改了



vm.\_data.student === vm.student true because数据代理

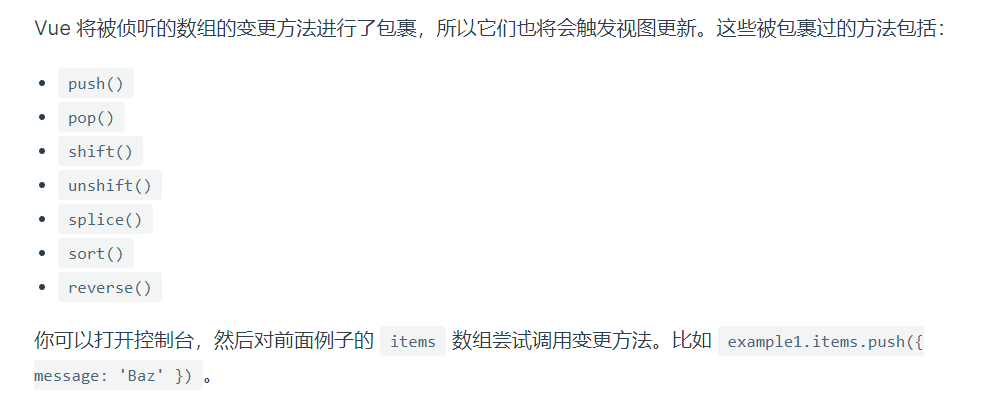


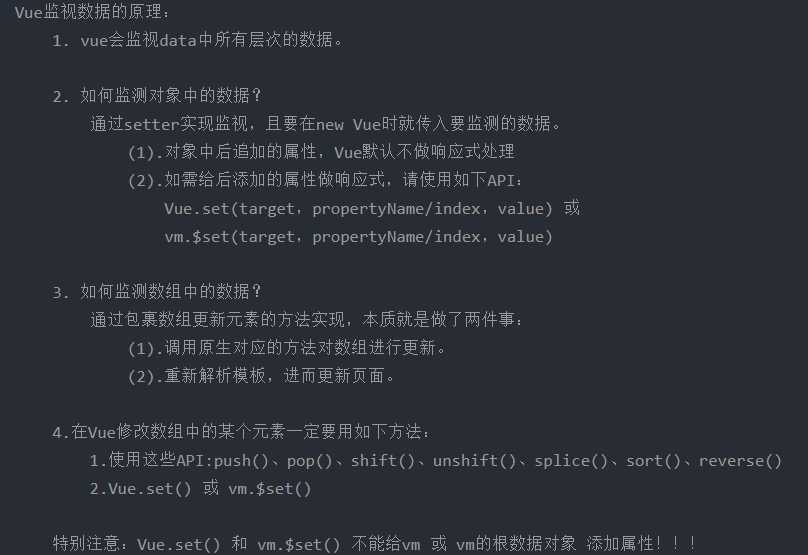
这两种写法一样

vm的根数据不允许做set的参数，比如data，但是\_date.xxx可以

数组的元素不会去匹配getter和setter而对象里的元素会

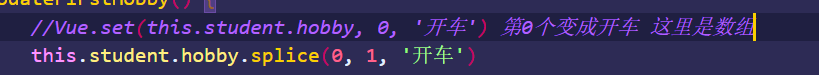
数组改了是改了的，但是vue不认，也就是检测不到





unshift往前加一个



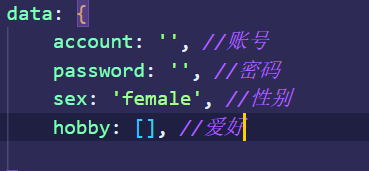


**所有不是由VUE所控制的回调，尽可能的写成箭头函数，原因是箭头函数没有this会向上找，找到vm**

v-model默认收集输入框的value值所以才能简写成v-model

所以单选或者多选一 写个value值即可

多选checkbox 要指定value并且hobby写成数组



不同场景使用的不一样

表单里的按钮点了默认跳转或者刷新页面

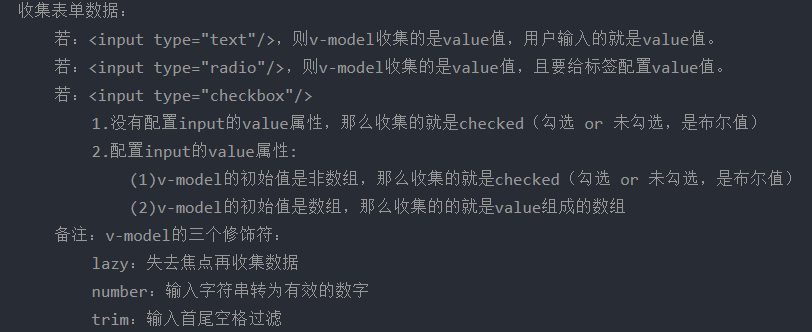




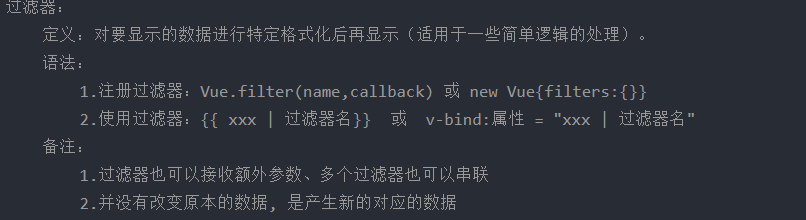
.lazy输入完了才收集



trim去掉前后空格



全局是在newvue之前写的





效果一样 都是输出尚硅谷

v-text会拿到name的值替换掉整个div里的内容，原先有的内容也被覆盖了

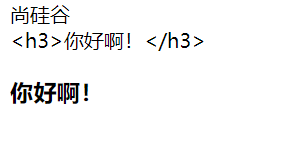
v-text把所有的字符串都当成文本去解析，也就是说标签也会被当成字符串，不会解析

如果换成v-html就会显示

来对比一下吧



上面是v-text，下面是v-html



cookie不能跨浏览器 ，比如你在谷歌登录了github，你在360打开是没有登录的，因为没有cookie

js代码拿到cookie

document.cookie就拿到了网站所有的cookie



往百度跳

百度写的是干坏事的服务器 俗称钓鱼网站



获取你当前所处网站的全部cookie然后跳走

并且把所有的cookie以query参数的形式传给你写的服务器，这里写的是百度

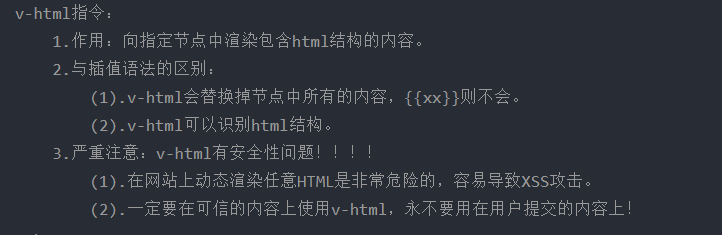


cookie跟在.com的后面，这里我没有弄所以没有

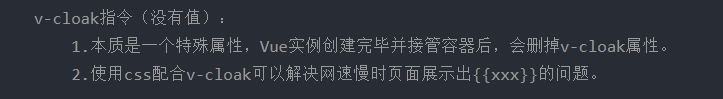
所谓盗号原理

httponly只有http协议能操作，js代码就不行了

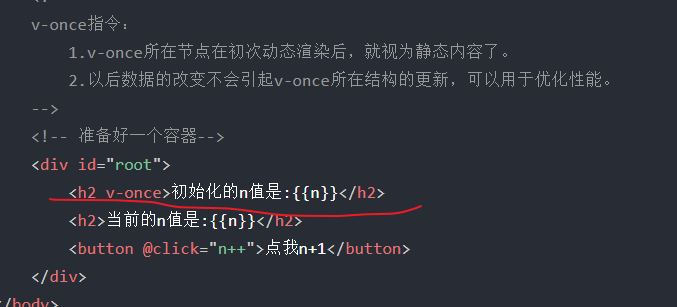
把标签解析了再用本身就是一种危险的行为

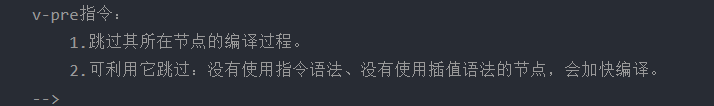






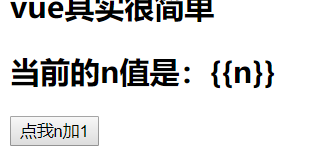
v-once没有值，只执行一次

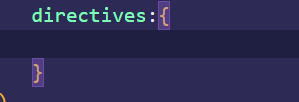




看效果

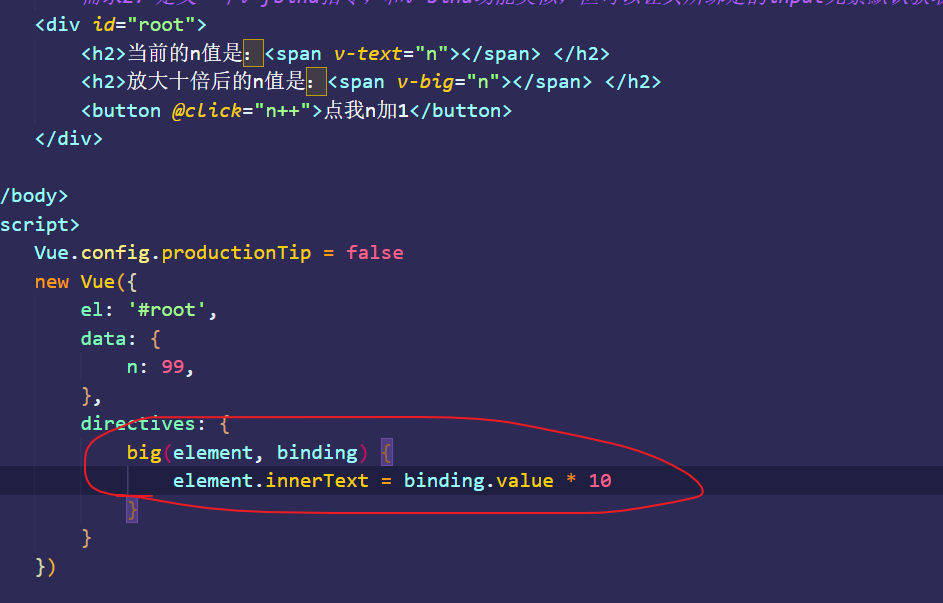






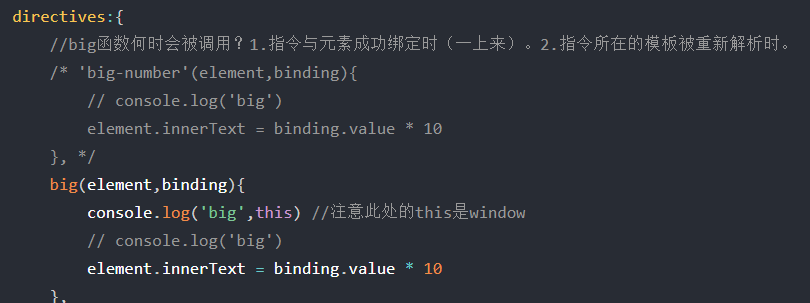
这个配置项是写自定义指令的

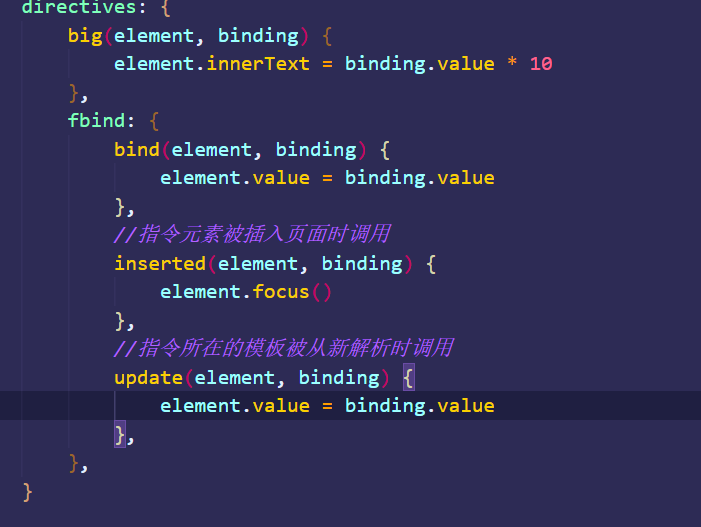
写的时候不用v-xxx， 用的时候才需要,直接xxx就行



这样就实现了

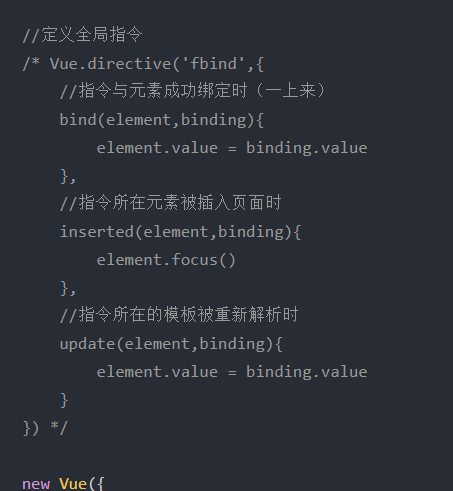
n变了big也会被调用，也就是会同时变化

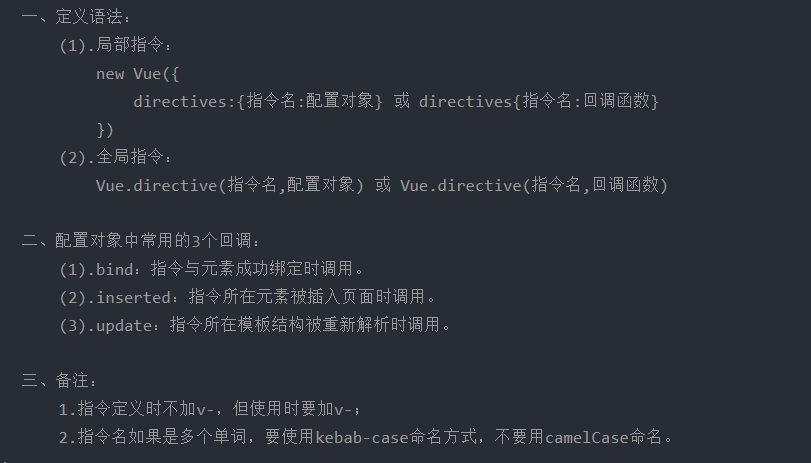




指令相关的回调函数里的this不是vm，而是window

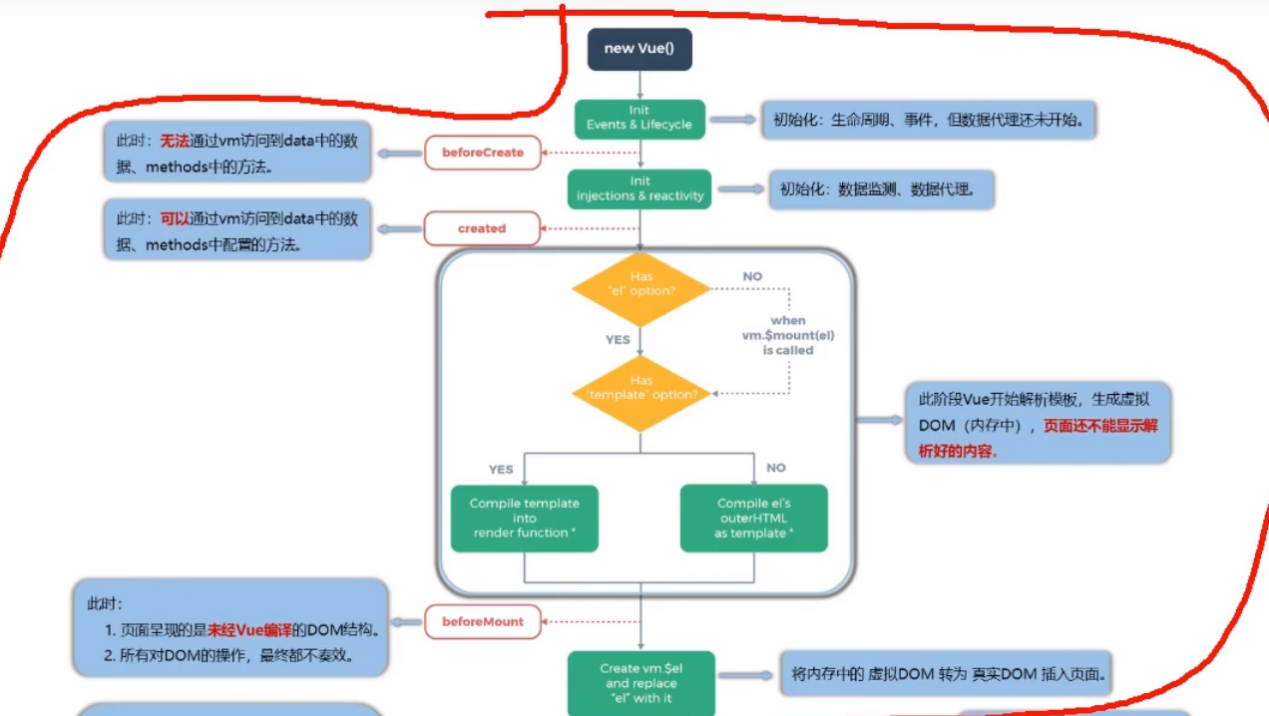
比如上面的big，.update等函数里的this都是window

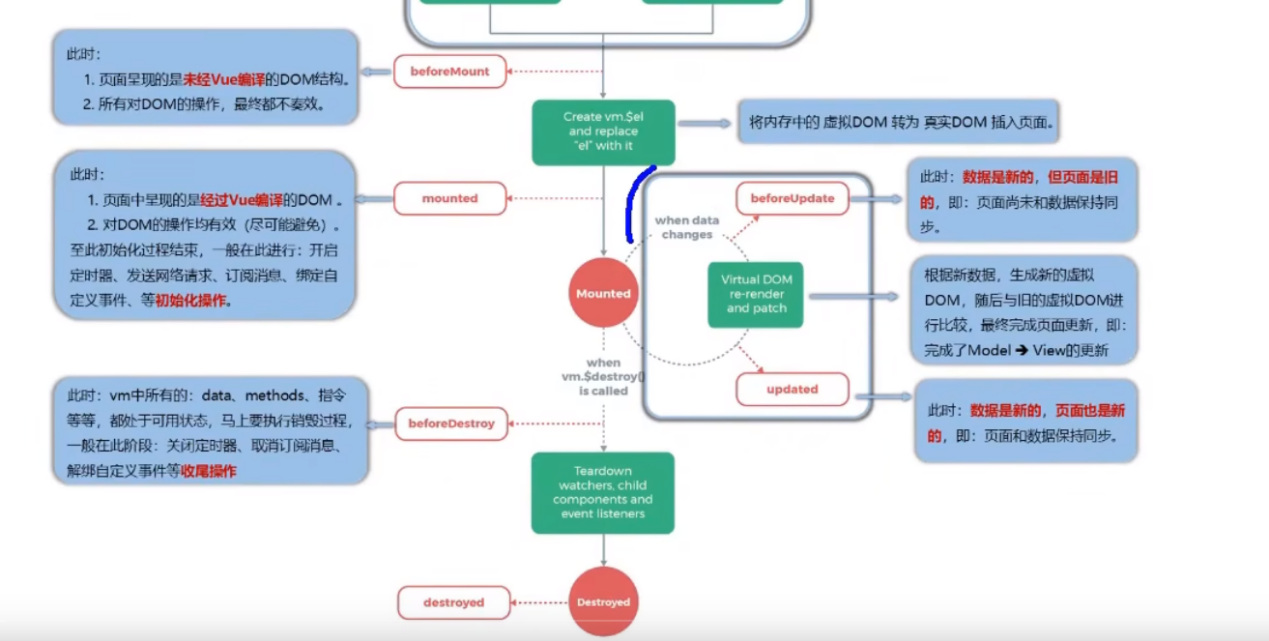


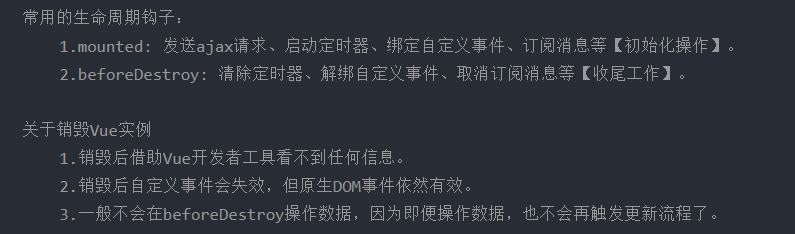


生命周期

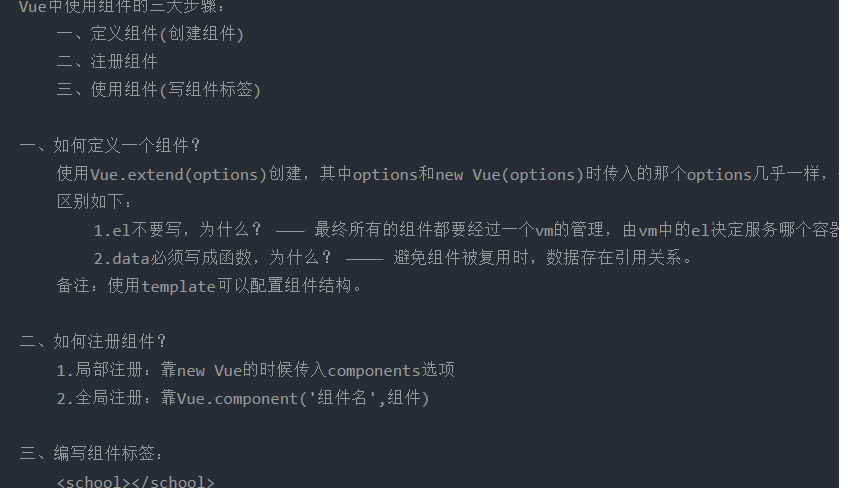
挂载流程到beforMount

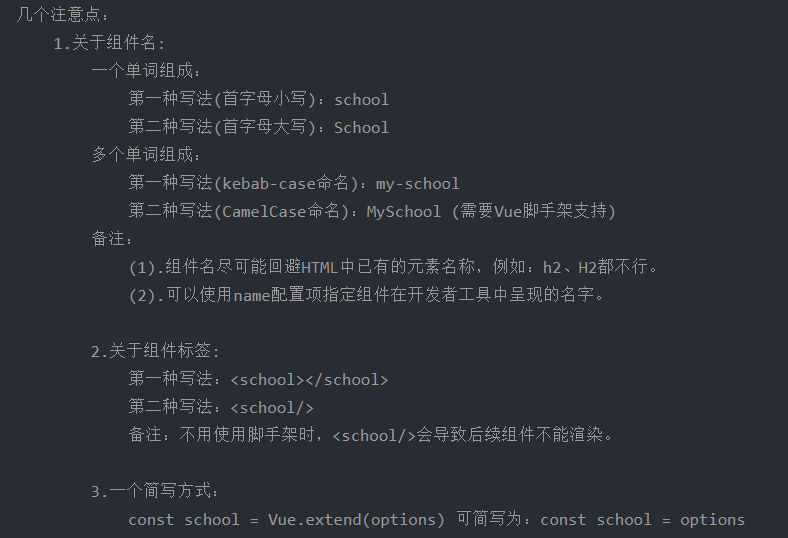






组件里的data不能写成对象式，只能写成函数式





关于VueComponent：

1.school组件本质是一个名为VueComponent的构造函数，且不是程序员定义的，是Vue.extend生成的。

2.我们只需要写<school/>或<school></school>，Vue解析时会帮我们创建school组件的实例对象，

即Vue帮我们执行的：new VueComponent(options)。

3.特别注意：每次调用Vue.extend，返回的都是一个全新的VueComponent！！！！

4.关于this指向：

(1).组件配置中：

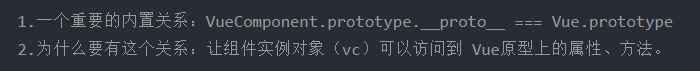
data函数、methods中的函数、watch中的函数、computed中的函数 它们的this均是【VueComponent实例对象】。

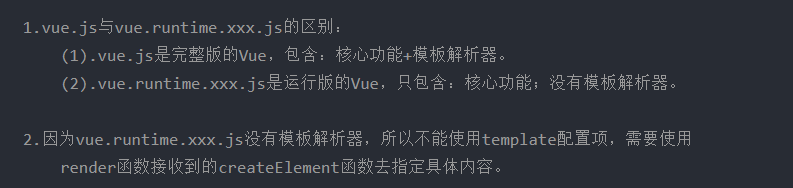
(2).new Vue(options)配置中：

data函数、methods中的函数、watch中的函数、computed中的函数 它们的this均是【Vue实例对象】。

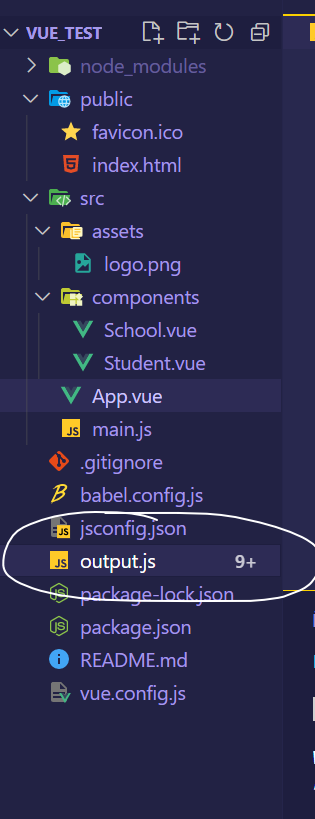
5.VueComponent的实例对象，以后简称vc（也可称之为：组件实例对象）。

Vue的实例对象，以后简称vm。

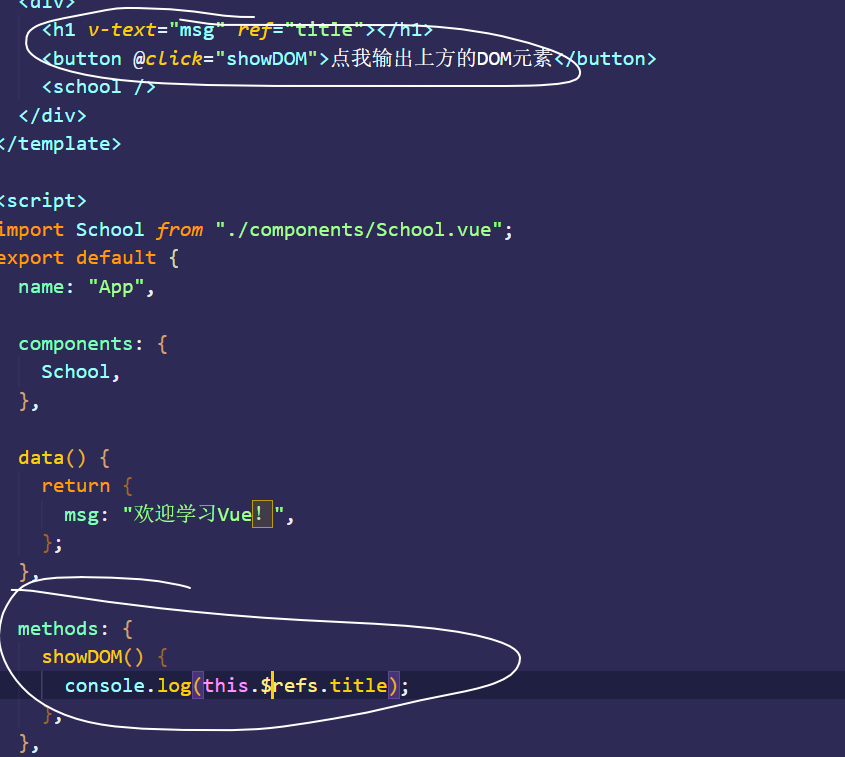




显示vue脚手架的隐藏文件夹

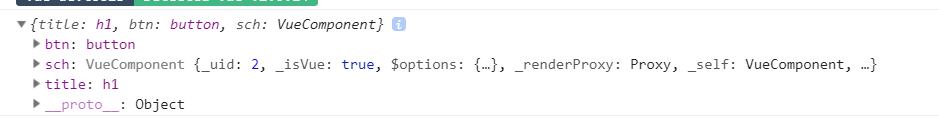


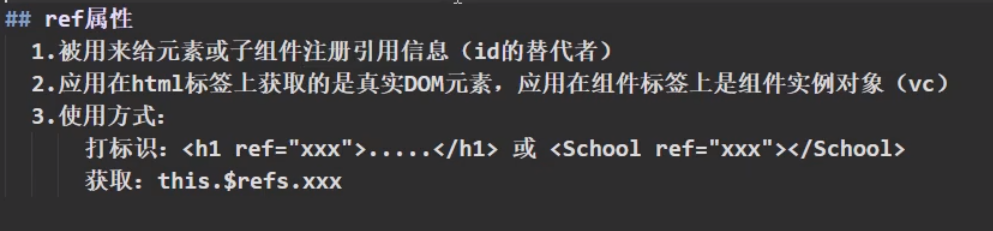
ref





也可以直接this.$refs拿到所有含ref的对象

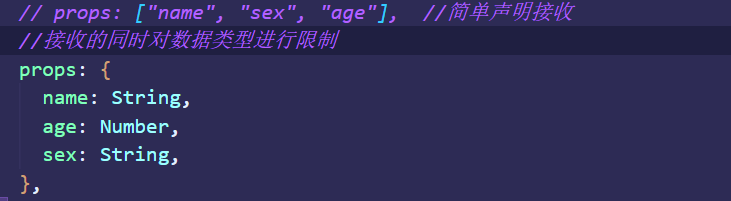






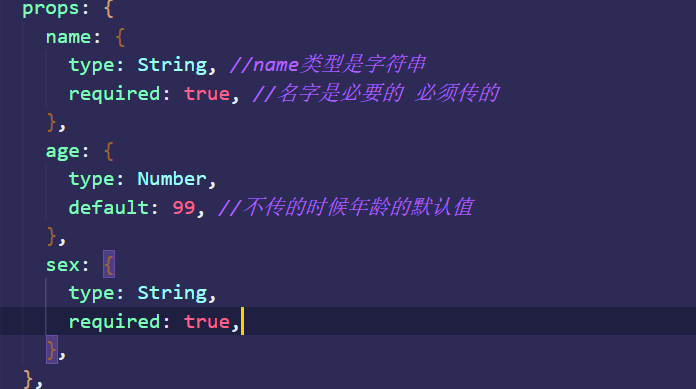
age前面加冒号执行“”里面的，不然就是字符串，+1就是字符串拼接会显示181

加了冒号后就显示19, 单向绑定



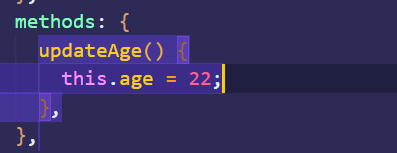
写成对象式 这样写age还是要加冒号

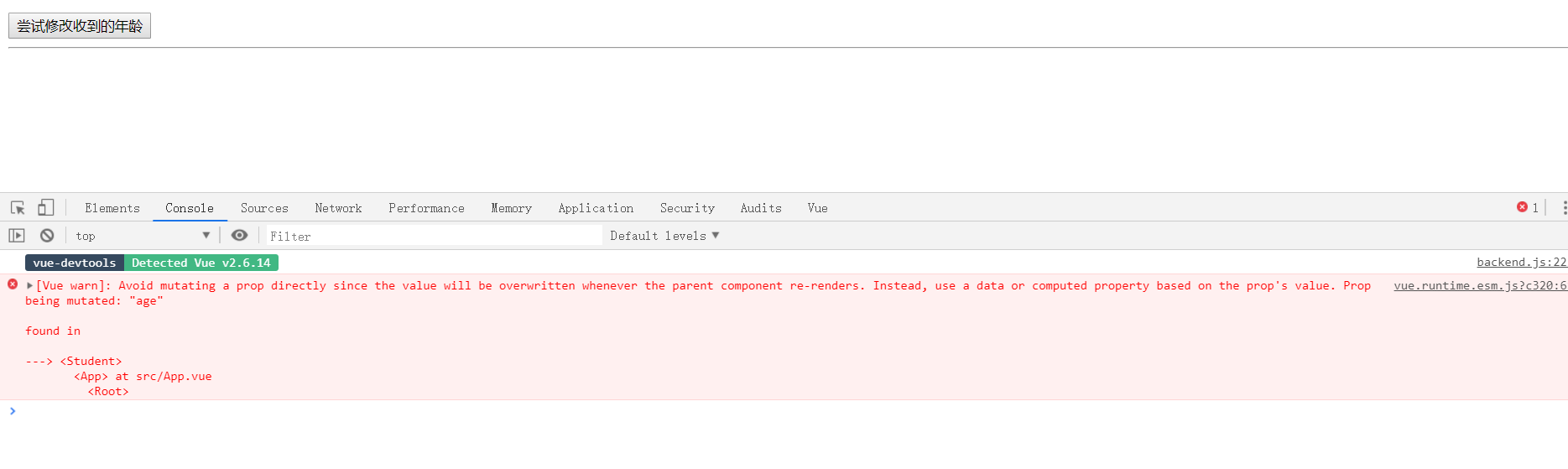
还有种写法 接收的同时对数据类型进行限制并且还有默认值的指定还有必要性的限制



required 跟 defaul一般不一起写

prop传进来的东西是不能改的，因为没有双向绑定





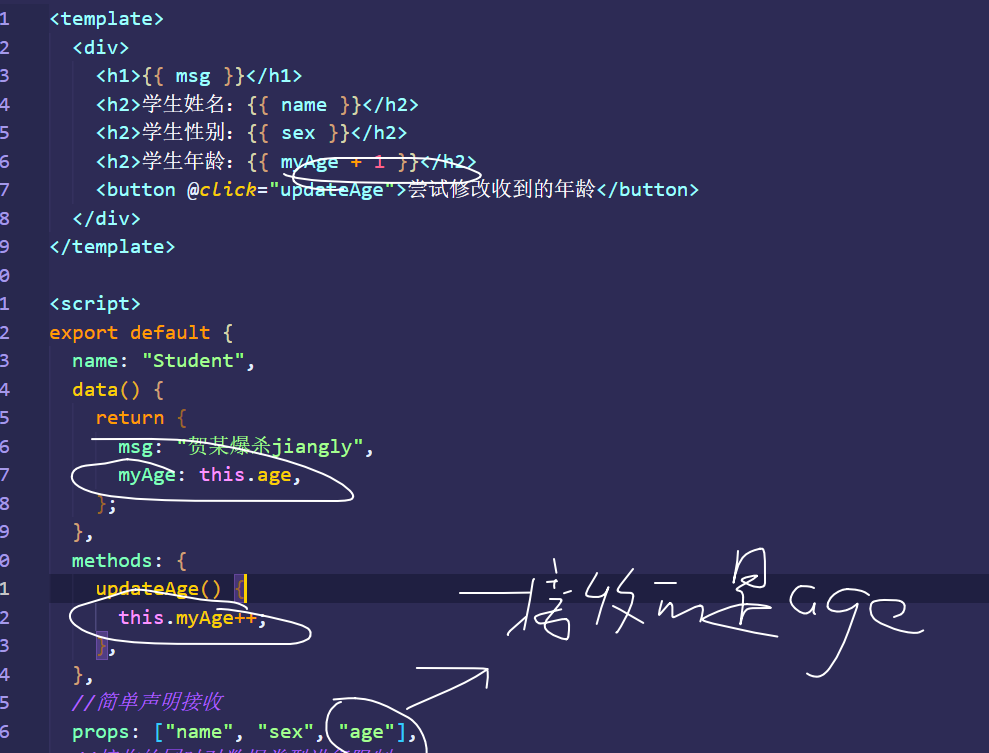
会报错 能改并且能有效果 但是尽可能不要改，因为会造成一些奇奇怪怪的问题

但是确实要改怎么办

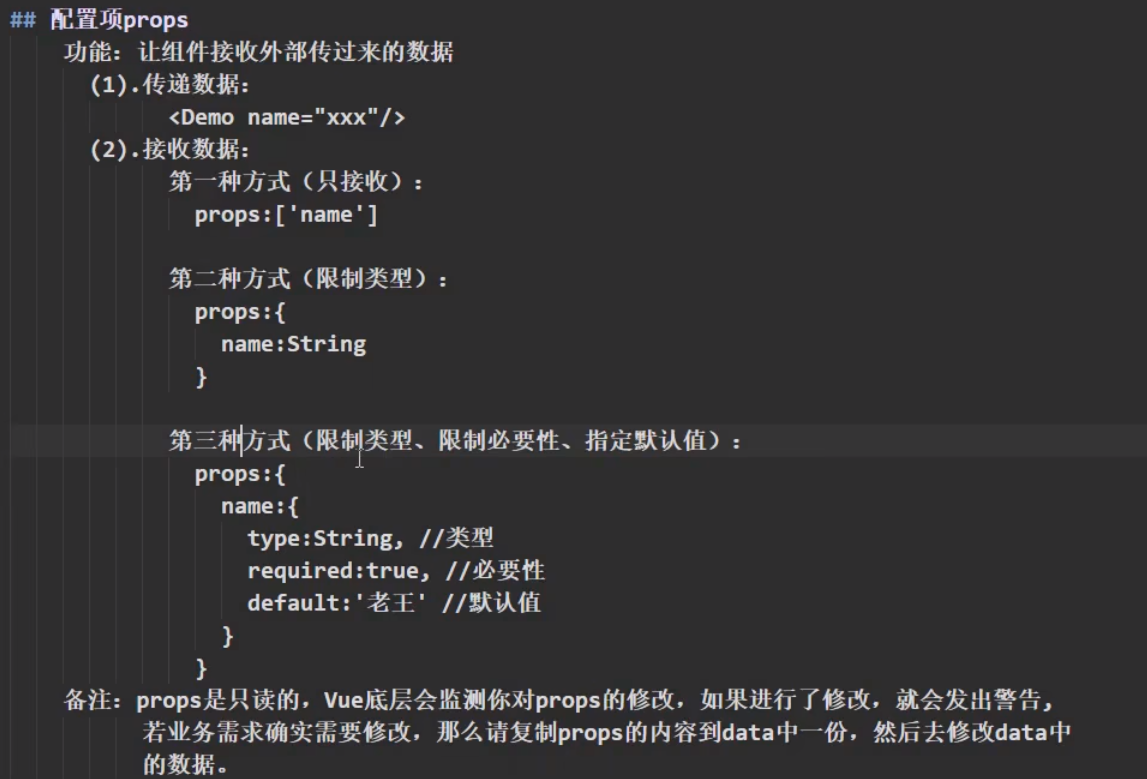
可以这么干

利用优先级

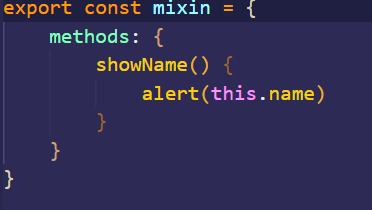
props里的东西是优先被接收的 优先被放在vc上的



但是不是什么都能传，关键字不能传 默认值也能传



export 分别暴露



用的时候这么用

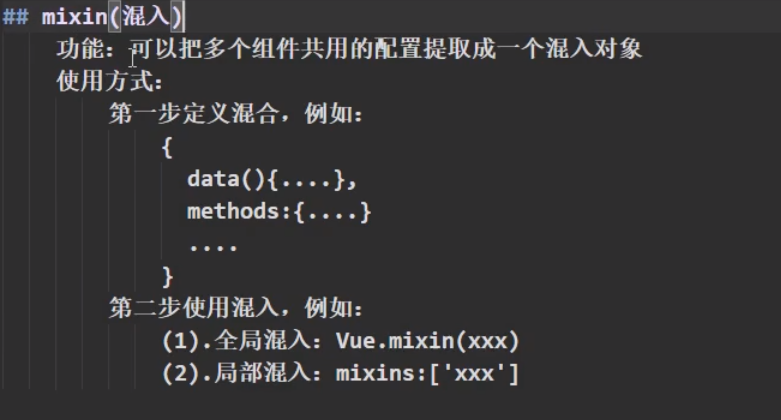
 {mixin}是const后面的名字

混合的配置项是mixins多了一个s哦

 两个一起复用

生命周期不以任何为主，冲突的话都会执行

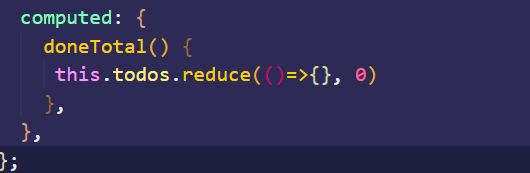
而且混合里的mounted先执行 数据的话以自己的为主，不以混合的为主



confirm弹窗有两个按钮，一个确认一个取消，确认的话就为真，否则为假

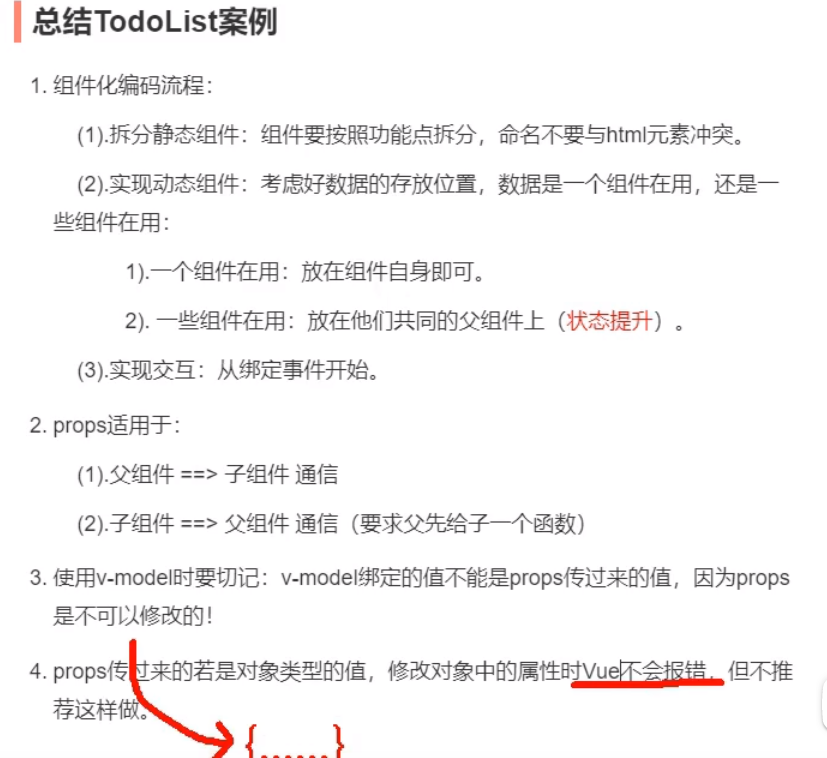
filter不改变原数组

reduce用作条件统计 第一个参数是一个函数，第二个参数是统计的初始值



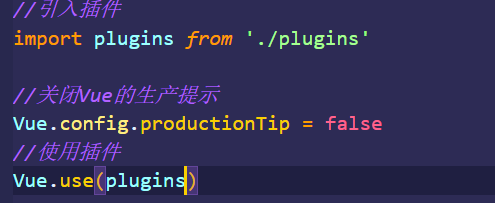


pre 和 current是上一次跟当前的值(这里current是todo对象) 最后一次调用reduce的返回值是整个函数的返回值



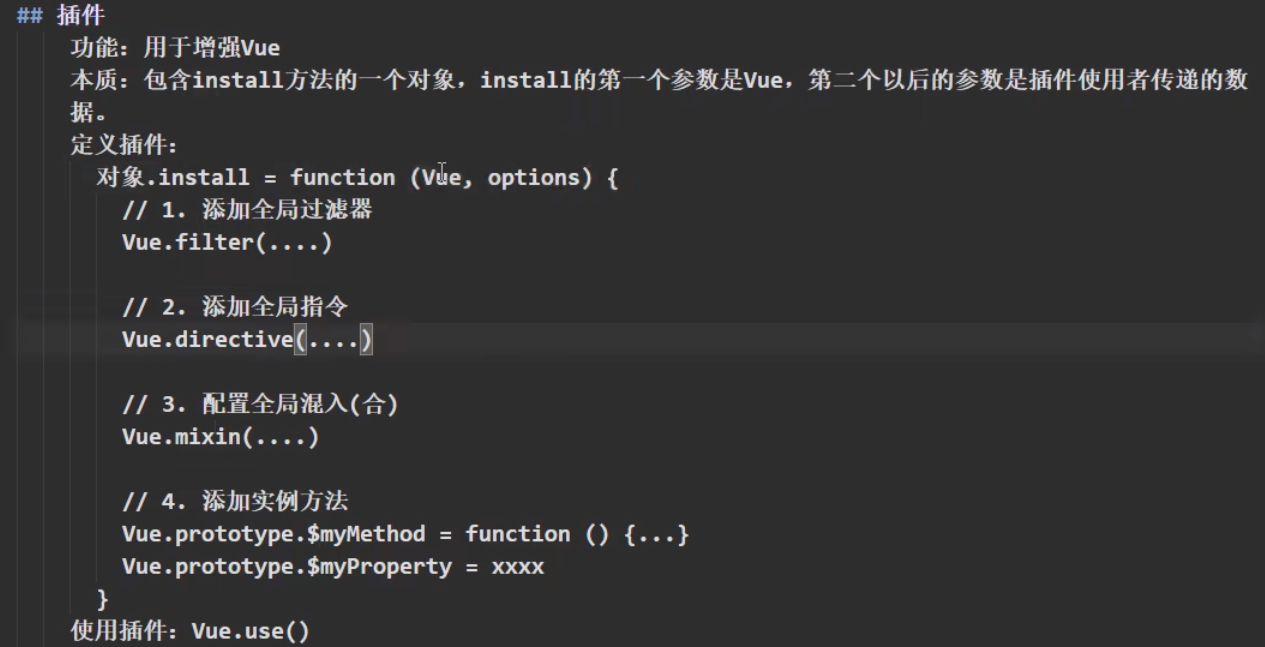


手写插件，一定要有install方法哦！ 用插件一定要先引入



*给Vue原型上添加一个方法（vm和vc就都能用了）*

引入和使用插件



scoped解决类名冲突等 因为会后来者居上（看谁先引入）

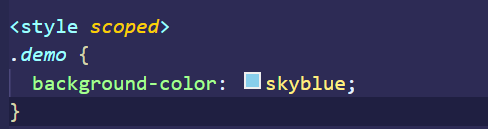
有个组件不太适合用 -app 为什么呢》

因为app是全局的 加了scope就无法应用到其它组件了

总结 scoped样式

作用：让样式在局部生效，防止冲突

写法 <style scoped>



checked默认都勾上

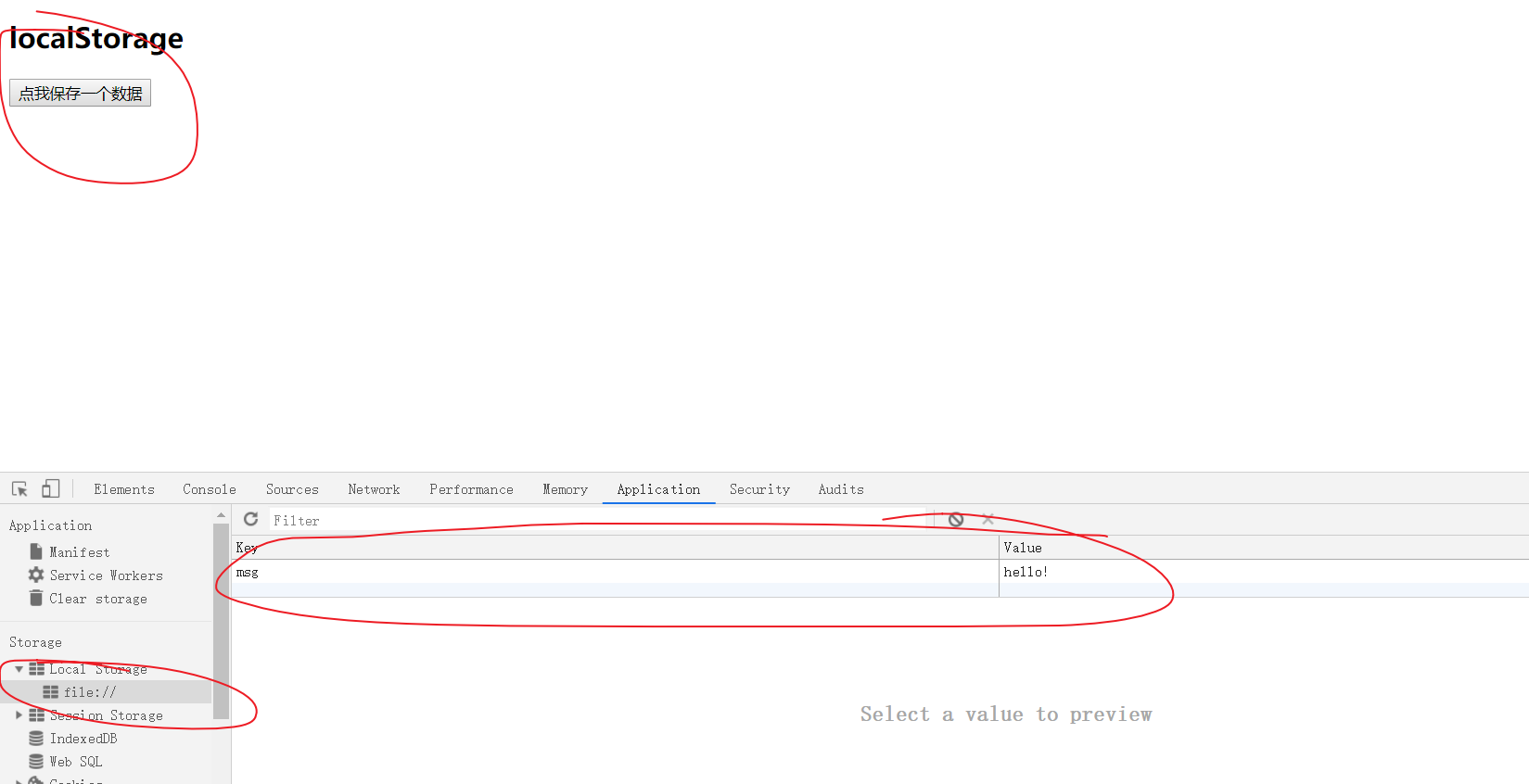
window可以直接写



可以改的，所以可以恶作剧哈哈哈



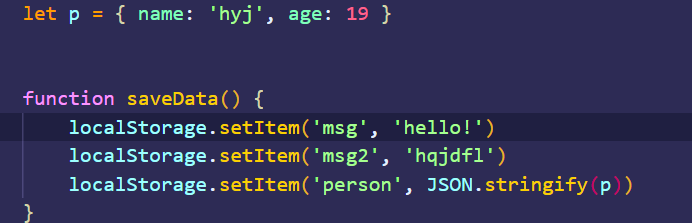
这么一写数据就保存进来了哦

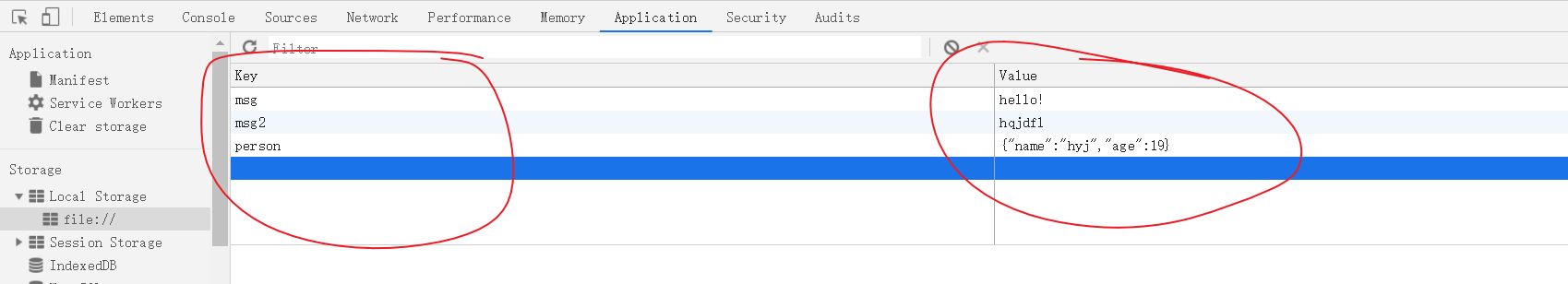


key-value都得是字符串

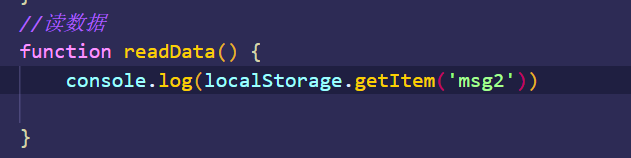
浏览器会帮你转换成字符串

JSON.stringify(p)把一个对象p变成字符串形式并且能体现里面的内容





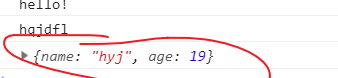
读数据 用getItem方法

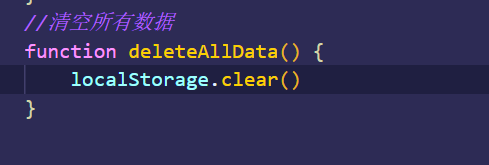
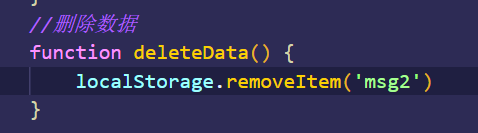


json.parse()方法将JSON格式字符串转换为js对象（属性名没有双引号）

用和没用







当读东西什么都没有的话是null

localStorage最大的特点是把浏览器关了也不会消失

sessionStorage是它的弟弟

sessionStorage最大的特点是把浏览器关了数据就会清空了

用户主动清除缓存也会在

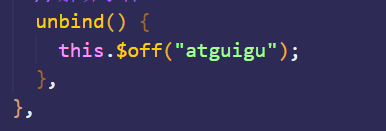


v-on绑定事件



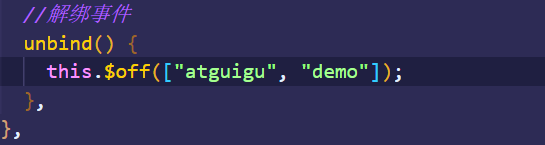


解绑事件用$off

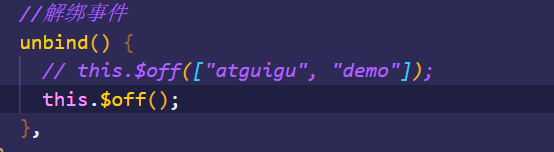


但是这种写法只适用于解绑一个自定义事件事件

解绑多个事件应该这么写写在一个数组里传两个参数



还有一种更加暴力的写法 这种是解绑所有的自定义事件



自定义的组件里的事件都被认为是自定义事件，除非是html里的标签，但是如果加上native就会被认为是vue原生事件



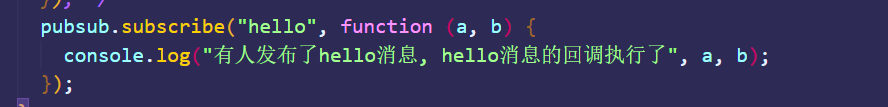




一挂载绑定最合适

需要数据的人要订阅消息

订阅 function两个参数分别是消息名（这里是hello），b才是数据



发布





也要记得取消订阅 跟$bus一样的



是通过id取消订阅的 不是名字

pubsub库里的this不保证是vc，所以function尽量写成箭头函数



@blur失去焦点事件



判断todo身上是否有isEidt属性

